

A Design of Resource Fault Handling Mechanism using Dynamic Resource Reallocation for the Resource and Job Management System

Young-Ho Kim, Eun-Ji Lim, Gyu-Il Cha, Seung-Jo Bae

Electronics and Telecommunications Research Institute

{kyh05,ejlim, gicha,sbae}@etri.re.kr

Abstract— Due to the development of NGS technologies and the reduction of analysis cost, it is possible to perform population-scale human genome analysis. Also, large amount of genome data have been exploded recently. It is required for introduction parallel processing using High Performance Computing systems to analyse and handle these large data through genome analysis pipeline. In this paper, we propose the resource fault handling mechanism based on dynamic resource reconfiguration and delayed scheduling for data-intensive pipeline job processing such as genome analysis executed on the large cluster systems interconnected by high speed and low latency network. In order to prevent the abnormal job completion caused by lack of the specific resources, we offer the resource fault detection and handling methods. If the cause of fault is lack of resources, it can be solved by the resource re-allocation and process freezing/resuming based delayed job execution or process migration on the available node.

Keywords— Resource management, Job scheduling, Resource fault, Process freezing/resuming

I. INTRODUCTION

Due to introduction of the Next-Generation Sequencing (NGS), it is possible that genome analysis cost for one person is less than \$1,000. As a result, the demand of large-scale bio data analysis and processing has been increased [1]. NGS technology is expected to obtain the genomic data of the individual level analysis and possibly accelerate in the realization of preventive care and personalized healthcare to provide information about the disease, medication. Accordingly, the analysis of human genome research with a focus on the NGS data has been actively proceeding. Genome analysis is large-scale data processing which is required a few terabytes(TB) data processing for the analysis goes through several steps one people. There are also jobs that require more processing time ten days on a single machine [2], [3]. The mainstream in the system configuration for the genome sequencing and analysis field cluster system consisting of a plurality of nodes rather than a single high-performance system. With the development of hardware technology, the type of resources that make up the cluster nodes become diversified and each resource is supported also having more

large capacity. In a cluster system environment with large computing resources, in order to utilize the resources efficiently, it is required a system for the management of computing resources and allocation of resources to do the job (job) submitted by the user in the optimal order and resources referred to as RJMS (Resource and Job Management System). Most of the high-performance computing systems are widely used the RJMS such as Torque, SGE (Sun Grid Engine), Maui [4]-[6]. Recently SLURM [7], which is the open source software system, is being used in many HPC systems about 60% of TOP 500 supercomputers including Tianhe-2, Sequoia, etc.

In the resource fault handling of conventional RJMS, sometimes abnormal termination happens for the specific job process which requests excess resource allocation more than allowed capacity in order to ensure other process's resource utilization even though entire system has enough available resources.

The rest of this paper is organized as follows. Section 2 looks at the related work and background of the genome analysis pipeline and resource and job management system. In Section 3 we present the resource fault handling approaches developed in our research. It describes the details proposed resource fault handling mechanism based on dynamic resource reallocation and process freezing/resuming. Consequently we conclude with future directions of work.

II. RELATED WORKS

A. Genome Analysis pipeline

Generally, a genome analysis pipeline for variation discovery goes through a series of analysis steps: read mapping, SAM-to-BAM format conversion, sorting of the mapped results, merging of the sorted results, and SNP detection, as shown in Fig. 1. The fragments are called read fragments or read sequences. The read fragments in the files are mapped to the reference genome sequence. This process is called read mapping or read alignment. For read mapping, various tools can be used; for example, BWA, SOAP, Bowtie, and so on, The mapping results are written in SAM format, which is a generic format for storing large nucleotide sequence alignments and is widely used in genome analysis

tools, Because the mapping results are unsorted and are often in multiple files, sorting and merging is required after alignment. Sorting is based on the mapped position of each read fragment in the reference genome. Moreover, for efficiency, the SAM format is converted into the BAM format before sorting. Finally, variants (that is, SNPs) are called in the SNP detection step (SNP detection is also called SNP calling). Additional steps such as realignment or quality recalibration can be positioned before the detection step to improve the accuracy of SNP calling.

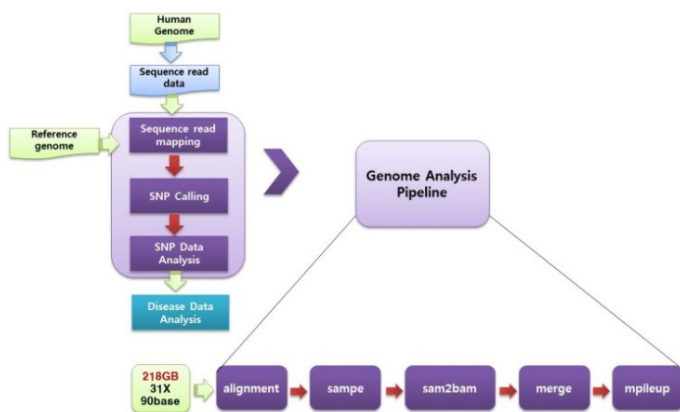


Figure 1. Genome analysis pipeline example

When genome analysis pipeline has been performed using BWA and SAM Tools, while it was executing the steps in sequence in a single node, we measured the resource statistics data for each system resources using ‘iostat’ tool. As a result of an analysis of the profiling of resources, it can be seen that profiling characteristics appears as shown in Table 1.

TABLE 1. CHARACTERISTICS OF PROFILING FOR GENOME ANALYSIS PIPELINE

Notation	CPU intensive	Memory intensive	IO intensive
Alignment	O	O	X
Sampe	X	O	O
Sam2bam	X	X	O
Merge	X	X	X
mpileup	O	X	X

B. RJMS Architecture

A typical architecture of RJMS is shown in Fig. 2. Generally RJMS is composed of four components: Queue Manager (QM), Job Scheduler (JS), Job Manager (JM), and Resource Manager (RM). Queue Manager accepts a job submitted by the user. The user must specify the required resources needed to perform each of these tasks during the job submission. The user may delete the job from the queue, and to search for information. The Queue Manager forwards job execution requests to the Job Scheduler. Job Scheduler sends

a resource allocation request for performing job process to the Resource Manager. Job Scheduler determine when, where, and how to execute job using the allocated node and resource information from the Resource Manager. Job Manager sends the job execution control to the agent of target node in order to execute the job process requested by the Job Scheduler with the allocation resources and execution parameters. Job Manager also gathers the execution status of running job process and sends the information to the Job Scheduler.

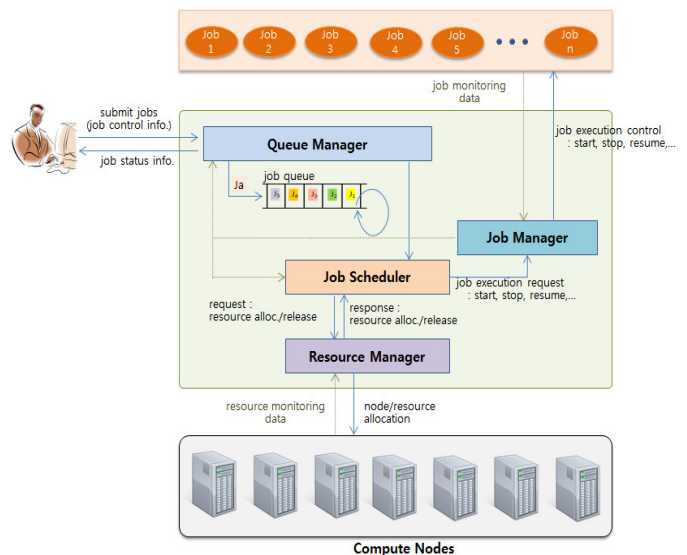


Figure 2. RJMS Architecture

III. RESOURCE FAULT HANDLING MECHANISM

In this paper, we propose a new system resource fault handling mechanism. It is achieved by dynamic reconfiguration and delayed job execution when running job process is going abnormal termination caused by resource allocation fault in HPC system environment. It is to provide a resource fault handling method which prevents the abnormal termination of the application due to the lack available resources. If the resource fault has been occurred on the running job process, it analyses the cause of fault and determines whether it is originated from resource-related fault.

If the cause of the fault due to the lack of memory resources, it stores the job process’s information of process execution context and resource allocation and stop the execution temporarily. It performs a re-allocation of scarce memory resources for the interrupted job and restores the suspended job process’s status and resumes it with reallocated resources environment.

A. Resource fault detection method

Fig. 3 shows a flow diagram of proposed system resources fault detection process. In the proposed detection method, if the resource related fault is occurred on the running job process, system fault handler routine is called. In a way that insert the inception code into memory fault handler included in the system fault handler routine, resource fault handler is performed before the memory fault handler is being worked.

Through interception (hooking) of memory fault handler, it generates an interrupt to the memory-related events occurring in the resource related fault by executing the 'resource_fault_handler' to handle that. In the proposed detection method, it is determined whether general page fault or resource fault caused by the lack of requested allocation memory is on the execution of 'pre_handler' triggered by interception code.

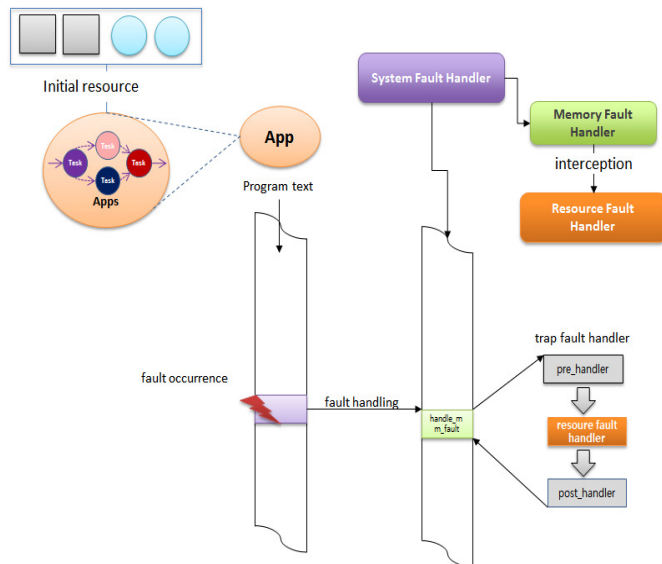


Figure 3. Resource Fault Detection Diagram

B. Resource fault analysis and handling

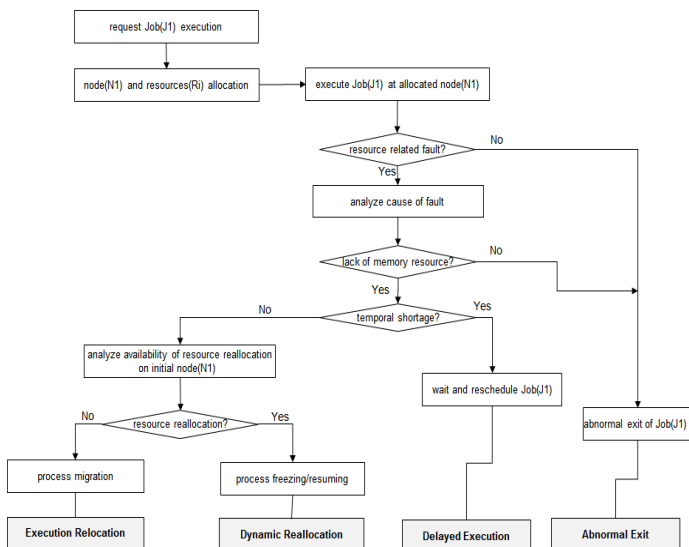


Figure 4. Flow chart of proposed resource fault handling

The proposed resource fault handling routine is worked as a flow chart in Fig. 4. If the cause of fault is lack of resources, first it saves the running job process's state by using process freezing technique. If the node has sufficient available resource and the problem can be solved through dynamic resource reconfiguration by increasing the amount of shortage resource, thereby suspended job process should be stored the context and status and resuming it. If dynamic resource

reconfiguration is impossible due to lack of available resource in the same node, first find another available node which has allocated enough resources, and then transfer job process's context and resource status to the newly assigned node and resume the new job process from restored job state.

In the phase of cause analysis of the resource fault, it is carried out based on the resource usage information about execution job process and system node. It is determined by using collected information from the Resource Manager whether the dynamic reconfiguration is needed.

TABLE 2. METHODS OF RESOURCE FAULT HANDLING

method	description
Delayed Execution	Due to temporary lack of resources, job is resumed after resource is available
Dynamic Reallocation	Initial resource allocation for job is not enough, but job can be resumed through resource reallocation by additional resource allocation
Execution Relocation	Due to lack of node resource, resource can't be reallocated, job should be resumed on other available node through process migration
Abnormal Exit	Resource related fault except resource allocation, job may be abnormal exit

If resource fault has been occurred, depending on the source of the fault and the status of system and job's available resources, it can be handled as following four ways.

TABLE 3. PARAMETERS RELATED WITH RESOURCE FAULT HANDLING

Notation	Description
R_i	initial resource assignment for Job process
R_u	current resource usage of Job process
R_e	allocation request amount by Job process
R_n	Total resource amount of Node
R_a	total allocation of available resources within a Node

1) Delayed Execution:

In the situation of Job (J1) is allocated smaller than the allowed resource capacity, it is requested lower capacity resource than that of available amount of node(N1) resources. The resource allocation request of J1 could not be processed even though it does not exceed the allowed resource amount for J1. It is a temporary shortage of system node resources. It can be handled through the delayed execution at the time that system resource is available. The resource condition of delayed execution can be represented as equation (1).

$$R_u + R_e \leq R_a \text{ and } R_a \geq R_e \quad (1)$$

2) Dynamic Allocation:

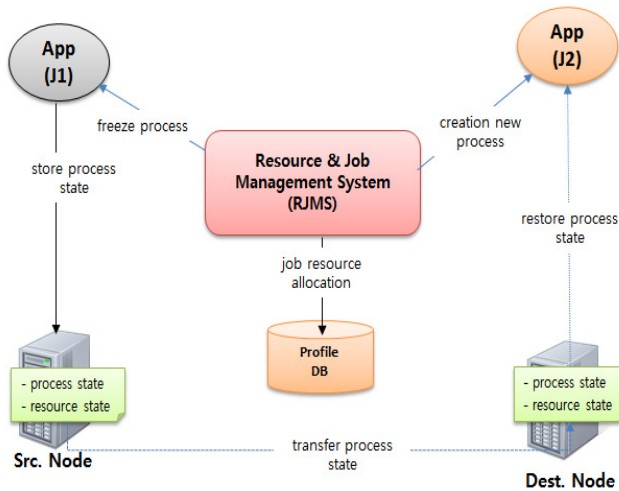
In the case that node (N1) has enough resource capacity, so dynamic resource configuration is possible through additional allocation of faulted resource. The requested allocation resource amount is larger than permitted for Job (J1).

However the resource capacity of N1 can afford to allocate additional resource amount of J1. It is required to reconfigure the resource of J1 by dynamic resource allocation corresponding to allocation request for larger capacity than allowed capacities (R_i) J1. And then, it resumes from the interrupted point of Job process. The resource condition of dynamic allocation can be represented as equation (2).

$$R_u + R_e \leq R_i \text{ and } R_n - \sum_1^n R_i \geq R_e \quad (2)$$

3) Execution Relocation:

In the case that the amount of allocation request is larger than allowed resource capacity to the Job (J1), node (N1) cannot afford to allocate additional request for Job J1. Because it is impossible to allocate in the same node due to over allocation request, new node N2 which has enough resources to allocate for interrupted Job J1 with initial allocation and additional capacity is selected by Job Scheduler. And then it creates new process J2 in node N2 to migrate interrupted process J1 from node N1. It restores from transferred J1's process context and resource information and resumes new process J2 from the freezing point of J1 as



illustrated in Fig. 5.

Figure 5. Flow Diagram of Process Migration caused by Execution Relocation

The resource condition of execution relocation can be represented as equation (3).

$$R_u + R_e \leq R_i \text{ and } R_n - \sum_1^n R_i < R_e \quad (3)$$

If it is impossible to allocate resource in the same node, new node assignment and resource allocation is necessary. And then, process migration is processed from the job J1 on original node N1 to new job process J2 on destination node N2. It transfers all suspended process status data and restores to resume new process. Details relating to the status of the process and resources are given in Table 2.

TABLE 4. PROCESS'S STATUS AND RESOURCE DATA

Item	Description	Details
------	-------------	---------

process's task_struct	process's information task	user_id group_id process priority process state process id etc.
process's mm_struct	memory map structure	pointer of page directory VMA's start/end address
process address space	range from 0 to 4GB	stack data code heap
open files	refers files by file descriptors	regular files pipes sockets
current data	Current information status	current working directory current root signals

IV. CONCLUSIONS

In this paper, we propose an efficient system resource fault handling mechanism using dynamic resource reconfiguration and process freezing/restart in HPC cluster system composed of various system resources. In an environment that job processes composed of NGS pipeline with a plurality of computing nodes and resources assigned by RJMS, it prevent abnormal termination due to system fault caused by lack of the specific resource. Through interception of system fault handler routine, we perform detection and handling of system resource fault. Applying the proposed resource fault handling mechanism, it is possible to prevent abnormal termination of job process executed in the long time NGS pipeline due to the under estimated resource allocation by the users, so it increase the efficiency of system resource utilization. It also reduces the time cost of additional work according to the re-execution of time-consuming job process caused by resource allocation fault.

In the future, we have plan to apply the proposed resource fault handling mechanism onto open source SW RJMS, SLURM. We will implement and experiment with various real NGS pipeline through the occurrence of resource allocation fault by selected fault injection provided by Linux Fault Injection Tool.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.B0101-15-0104, The Development of Supercomputing System for the Genome Analysis).

REFERENCES

- [1] L. Stein, "The Case for Cloud Computing in Genome Informatics", *Genome Biology*, vol. 5, no. 11, 2010.
- [2] Human Genome Project, Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Human_Genome_Project

- [3] Yunku Yeu et al., "A survey of sequence alignment algorithms for next-generation sequencing read", *KISE Database Society Journal*, vol.28 no.1 pp.33-51, 2012.
- [4] G. Staples, "Torque resource manager," in *Proceedings of SC'06*, 2006.
- [5] Sun Microsystems, Inc. Sun grid engine. [Online]. Available: <http://gridengine.sunsource.net/>
- [6] D. Jackson, Q. Snell, , and M. Clement. "Core algorithms of the Maui scheduler". In *Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science. Springer-Verlag, 2001.
- [7] Slurm Workload Manager, SchedMD, [Online]. Available: <http://slurm.schedmd.com/slurm.html>



Young-Ho Kim was born in South Korea in 1973. He received the B.E., M.E. degree in Information and Communication Engineering from Chungbuk National University, Cheongju, Korea, in 1999, 2001, respectively.

He joined Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea, in 2001. Since 2001, he has been with the cloud computing department, where he is currently a senior research member of engineering staff. His main areas of

research interest are High Performance Computing, Cloud Computing, System Management, and Distributed Computing Systems.



Eun-Ji Lim received the B.E., M.E. degree in Computer Science from Pusan National University, Busan, Korea, in 1999, 2001, respectively. Since 2001, she has been with Cloud Computing Department in Electronics and Telecommunications

Research Institute (ETRI), Korea, where she is currently a senior researcher. Her main areas of research interest are Distributed System and High Performance Computing



Gyu-Il Cha was born in South Korea in 1970. He received the B.S., M.S. degree in Computer Science from Korea University, Seoul, Korea, in 1998, 2000, respectively.

He joined Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea, in 2000. Since 2011, he has been with High-Performance Computing Research Section, where he is currently a senior research member of engineering staff. His

main areas of research interest are High Performance Computing (HPC), System Architecture, and Kernel software.



Seung-Jo Bae received his MS degree in Computer Science and Ph.D. degree in Computer & Information Science from Syracuse University in 1992 and 1997, respectively. He is a principal research scientist at Electronics and Telecommunications Research Institute (ETRI) in Korea. His research interests are in the area of High Performance Computing & Parallel Computing.